Cromemco Software Update Service Note CDOS-1

Date: June 4, 1982

Product: DOS-L and DOS-S

Version.Release: 02.52

Date production of this version began: June 4, 1982 on 8"
June 4, 1982 on 5"

First serial number with this version: D01210380 on 8"
D01311246 on 5"

[Note: Earlier serial numbers may also contain this version. All Cromemco Finished Goods stock was recopied on the above dates.]

SUMMARY

This release of the Cromemco Disk Operating System, CDOS, includes several enhancements over version 2.36. CDOS now supports 5" hard disks and the new **WDI-II** disk controller. Several new system calls have been added to CDOS version 02.52. This update note also covers modifications to the intrinsic command **TYPE** and the utilities **XFER, DUMP, STAT,** and **INIT.**

5 INCH HARD DISK AND CONTROLLER

Version 02.52 of CDOS contains a device driver for the new **WDI-II** hard disk controller which is used in conjunction with the new 5" hard disks as well as the 8" hard disks. The new 5" hard disks will run only with the WDI-II controller. CDOSGEN now configures versions of CDOS for use with 6 or 11 megabyte capacity disk drives. The 5" hard disks hold 6 megabytes; 8" hard disks hold 11 megabytes.

TYPE COMMAND

A minor modification to the **TYPE** command improves the way in which this command interfaces with the Cromemco 3102 terminal. TYPE no longer sends to the console the escape character (ASCII 1B) and CNTRL-E (ASCII 05). This modification helps prevent the Cromemco 3102 terminal from entering undesirable modes when these characters are sent to the console using the TYPE command.

XFER UTILITY

XFER, as described in the **CDOS Instruction Manual**, part number 023-0036, can expand tabs using the **/T.** It will also delete any null (00h) and delete (7Fh) ASCII characters when the **/S** switch is used. However, the user must specify that the original file is an ASCII file by using the **/A** switch with either (or both) of **/T** or **/S** switches.

The current version of CDOS can write and read random files. Changes in XFER allow it to handle random files as well. Switches which have been added or changed are:

XFER/C/X FILE1=FILE2

This command displays the address and contents of differing bytes in the two files.

XFER/C/B FILE1=FILE2

This command returns the range of any discrepancies found in the two files. The range is described by two values. The first value is the hexadecimal location where the difference starts. The second value is the hexadecimal location +1 of the last difference.

XFER/A/I FILE1=FILE2

This command ignores imbedded CNTRL-Zs in file2 and does not pass them to file1.

XFER accepts expanded ambiguous file names on the command line. This change also applies to STAT described below. The command can now accept an asterisk, *, to mean any ambiguous string. The brackets, [], contain a list of characters which can occupy that position in the string. Ranges are specified

by listing the first and last characters of the range separated by a dash. An example of the use of ambiguous file names is:

```
XFER/V D:=A:*A.Z80 [A-DH]*.[RZ]*
```

The above statement tells XFER to transfer those files on disk A to disk D which have the following characteristics:

The files should begin with any string sequence, followed by an **A** before a dot, and a **Z80** extension. Files which begin with **A, B, C, D,** or **H,** followed by any string, a dot, and an extension which begins with **R** or **Z,** followed by any string should also be transferred.

XFER can accept arguments which are separated by commas or spaces. For example:

```
XFER B:=A:NEW NEWER NEWEST
XFER B:=A:OLDEST,OLDER,OLD
```

The first form, which separates the three NEW arguments with spaces, will copy the file A:NEW to the file named B:NEW. Likewise, A:NEWER and A:NEWEST will be duplicated on disk B with the appropriate names NEWER and NEWEST.

The second form, which separates the three OLD arguments with a comma, will copy the contents of A:OLDEST to disk B under the file name of OLDEST. Next, XFER appends the contents of A:OLDER to the contents of B:OLDEST. Then the contents of A:OLD will be appended to the contents of B:OLDEST. The comma-separation form concatenates three distinct files on the source disk into one distinct file on the destination disk. Another example:

```
XFER/A/I LARGE=SMALLEST,SMALLER,SMALL
```

The above command concatenates SMALLEST, SMALLER, and SMALL into one file called LARGE. It will also ignore all CNTRL-Zs in the first two files, SMALLEST and SMALLER.

It is possible to print on an alternate printer by the command

```
XFER PRT:2=fILE
```

Only one random file may be specified as an argument to XFER.

Console Port Addresses

Console I/O ports 0 through 7 have the following hardware port addresses:

Console 0 = 00h
Console 1 = 20h
Console 2 = 50h
Console 3 = 60h
Console 4 = 70h
Console 5 = 80h
Console 6 = 90h
Console 7 = F0h

A default device can be given in location F0h.

DUMP UTILITY

DUMP version 2.02 gives users six new switches making the CDOS DUMP utility as powerful as the Cromix Operating System Dump utility. The table below lists the switches and corresponding uses. Two of the switches allow the user to specify the beginning or ending byte of a file for efficient, concise memory dumps. Another new switch is called the swath width. The definition of a dump's swath width is the ending byte number minus the beginning byte number. Users who know how far into the file they wish to go may specify the beginning byte and the swath width. Another new switch dumps records. If a file contains random or sequential records, users can specify the first record to dump.

DUMP has an option to dump read protected files. It is also possible to dump a file and add an offset number to the byte addresses. This may be desirable when dumping executable files which begin execution at 100h. An offset of 100h would give the displayed contents the same address as they would have when the file is in memory during execution. In general, DUMP displays a minimum of one block and additional bytes in whole block increments.

Format

DUMP/S1/S2... D:FILENAME.EXT

where

S1,S2,... are switches and
D is a disk specifier.

Switches

/B:XXXXXX
Begin the first block at byte XXXXXX.

/E:XXXXXX
End with the block containing byte XXXXXX.

/S:XXXXXX
Swath width

/R:XXXXXX
First record to be dumped

/R
Dump a read protected file

/O:XXXXXX
Address offset

In all cases, **XXXXXX** is a hexadecimal number.

Only one random file at a time may be specified as an argument to the DUMP utility. The following are some examples:

DUMP/B:400 C:FOREST

This command will dump the file FOREST on the disk in drive C beginning with the block containing byte 400 (in hexadecimal).

DUMP/R:2BA INVNTRY

This command will dump the file INVNTRY beginning with record number 2BA.

DUMP/O:100 ADDALL.COM

This command will dump the contents of ADDALL.COM so that the first byte of ADDALL.COM is byte 100 instead of byte 0. Thus, the dumping of a file can be offset to simulate another display such as **DEBUG.**

STAT UTILITY

Up to eight consoles may be in the drivers with this version of CDOS. STAT can be used to change a printer or console by a command such as:

STAT PRT:=2

STAT now accepts expanded ambiguous file references on the command line (see XFER above). For example,

STAT/A *A.Z80 [A-CF]*.[RZ]*

means

> Give status on all files beginning with any string followed by an **A**, a dot, and a **Z80** extension. Include files beginning with **A**, **B**, **C**, or **F**, followed by any string, a dot, followed by either an **R** or **Z**, followed by any string.

Notice that ambiguous file specifiers are separated by spaces, not commas.

INIT UTILITY

The INIT program is used to initialize diskettes for use with the computer system. The process of initializing a diskette consists of several stages:

1. Testing the diskette and drive for proper operation.
2. Formatting the diskette.
3. Writing certain information on the diskette to inform the operating system what type of diskette it is.
4. Writing information on the diskette to enable the operating system to store and access files.
5. Writing information on the diskette enabling the system to boot itself automatically.
6. Declaring alternate tracks for hard disk tracks with hard errors.

Not all of these steps are performed in all cases nor are they all performed by the INIT program.

Below is a sample script of a typical INIT session to format a 5" CDOS floppy diskette. The messages and questions are displayed by the INIT program. The user's responses are shown in bold face type. Following this are brief descriptions of each part of the execution.

```
Initialize Disks version xx.yy

Press:  RETURN to supply default answers
        ESC to abort formatting
        CNTRL-C to abort program
Warning:  INIT can destroy all disk data

Disk to initialize (A, B, C, D)? B

Testing:
 Index pulses being received correctly
 Rotational speed:  300 RPM

Formatting:
 Disk type (C=CDOS, X=Cromix)?  <C> RETURN
 Single or double sided (S/D)?  <D> RETURN
 Single or double density (S/D)?  <D> RETURN

 First cylinder (0-27H)?  <0H> RETURN
 Last cylinder (0-27H)?  <27H> RETURN
 Surfaces (0-1,All)?  <All> RETURN

 Cylinder, Surface:  restore
    00h, 0
    00h, 1
    01h, 0
    01h, 1
     :  :
     :  :
```

Disk to Initialize (A, B, C, D)?

The INIT program first asks for the drive letter of the drive containing the diskette to be formatted. Be sure to specify this letter correctly as the INIT program destroys all data on a diskette.

INIT will briefly test the specified drive to determine if it is operating correctly for diskette formatting. Drive speed is particularly important to correct formatting, so this is reported by the program. When formatting hard disks, the program also checks out the controller board hardware.

Disk Type (C=CDOS, X=Cromix)?

The next question asks whether you will be using the CDOS or Cromix Operating System with the diskette. You may select either the CDOS or Cromix Operating System, regardless of which system you are currently using to initialize the diskette.

Two more questions will be asked to determine what combination of sides and density are to be used.

First Cylinder (0-27H)?
Last Cylinder (0-27H)?

The INIT program asks for the numbers of the first and last cylinders being formatted. If the entire diskette is to be formatted, the operator should supply the default answers to these questions by pressing the RETURN key twice. Occasionally, it is necessary to format a portion of a diskette (e.g., one track that seems to have frequent soft errors), making it necessary for the program to ask these two questions.

Surfaces (0-1, All)?

The next question asks which surfaces are to be initialized. Usually, the default response of all surfaces is supplied by pressing RETURN. However, the user may choose to format only one surface by giving its number (one of the values shown in the prompt).

After the answers to all these questions have been supplied, the INIT program will proceed to format the diskette and will display its progress by giving cylinder and surface numbers. The diskette is always formatted from the outermost cylinders inward for proper head positioning.

Alternate Tracks

Since hard disks are designed for long term use and reliability, they have a provision for declaring alternate tracks. Alternate tracks are good tracks which may be used in place of tracks which, over time, develop hard errors. The locations of these tracks, once declared, are stored in a special area of track 0 (cylinder 0, surface 0) called the alternate track register. Once a hard disk has been formatted with INIT, the program will display a list of previously declared alternate tracks and will give the operator the chance either to change the tracks or declare new ones.

The display of alternate tracks will usually print either **cyl xx, surf y** to indicate the cylinder and surface numbers of the track with the hard errors, or will print **unassigned** to indicate that this alternate track has not yet been used. The message **illegal entry** is a warning that what has been stored in the alternate track register is an illegal value as far as the INIT program is able to determine. Generally, this should be changed to a legal declaration if one is known.

Following its display of the alternate tracks as originally defined, the INIT program asks two questions:

1. Does the operator wish to redeclare any of these already existing alternate track declarations?

2. Does the operator wish to declare any new alternate tracks?

It's usually best to leave the alternate track declarations for a drive as they were when the system was obtained from the factory unless there is a specific reason for changing them.

The INIT program cannot and should not be aborted during formatting or declaring alternate tracks of a hard disk as certain information about the drive is kept in volatile memory during this time. **Aborting the program will cause a loss of this information.**

If a decision is made to declare alternate tracks, the INIT program will prompt the user with the alternate track number and ask the user to supply the cylinder and surface numbers of the track with the hard errors. Press RETURN to leave the alternate track declaration as it was upon entering INIT. Type **U** (for **unassigned**) to free up a previously declared alternate track.

The INIT program makes no attempt to salvage the data stored on a track with hard errors which has been declared as an alternate track. It is generally best to recover as much of this data as possible prior to declaring the alternate track. The HDIAG program supplied as part of the Cromemco Diagnostics Software (CDS) package has special provisions for recovering this data. However, this procedure is complex and should be performed by an authorized Cromemco Dealer.

INIT ERROR MESSAGES FOR THE CDOS AND CROMIX OPERATING SYSTEMS

The following error messages may appear during the INIT program:

Incompatible with operating system
Use single user or simulator CDOS xx.yy or higher

This version of the INIT program is being used with an earlier, noncompatible version of CDOS or the CDOS simulator program (when running under the Cromix Operating System). Use the version of CDOS specified in the error message.

Initialization inhibited in this machine

Switch 4 of the 16FDC or 4FDC disk controller board has been turned on by the operator. This prevents disk initialization in this computer system.

4FDC not capable of double density operation

The operator has attempted to initialize a double density floppy diskette in a computer system containing the hardware for single density operation only (i.e., containing a 4FDC disk controller board).

Illegal value

The number supplied as an answer to a prompt is an illegal response for that question. Generally, this means the number is out of range.

Second number must equal or exceed first

This error message will be printed if the answer to the question **Last cylinder?** is not greater than or equal to the answer to the question **First cylinder?** The INIT program always formats disks from the outermost cylinders inward so as to provide for consistent head positioning.

Drive x is write-protected
Diskette in Drive x is write-protected

The specified hard or floppy disk has been write-protected and cannot be initialized until the drive or diskette has been write-enabled.

Drive x not ready

The specified drive is not ready, which generally means that it cannot be selected by the software. This usually occurs when a floppy diskette has been improperly inserted in its drive or the door has been improperly closed.

Can't Select Drive x, Status yy

This occurs for reasons similar to the above error. It usually indicates some type of hardware failure and reports the error status associated with the malfunction.

Can't Re-zero Drive x, Status yy

This error occurs with hard disk drives which cannot be rezeroed, or restored, without error. The error status associated with the malfunction is reported.

Init error: Drive x, Cylinder ww, Surface z, Status yy
Restore error: Drive x, Cylinder ww, Surface z, Status yy
Seek error: Drive x, Cylinder ww, Surface z, Status yy
Write error: Drive x, Cylinder ww, Surface z, Status yy

These messages indicate that an error has occurred on the specified drive, cylinder, and surface. The error status is reported. The meanings of the status values are identical to those reported by CDOS and Cromix error messages. These are explained in the June 1981 **CDOS Instruction Manual**, part number 023-0036.

Formatting aborted just prior to writing cylinder ww, surface z

Although not an error, this message indicates how much of the disk had been formatted just prior to initialization being aborted by the operator pressing ESCape.

PIO's not working
PIO's and direction control transceivers OK

The first of these error messages is printed when the parallel input/output channels of the WDI hard disk controller are not working correctly during the drive test phase of initialization. The second message will be printed informationally if the PIO's are working correctly as far as the INIT program is able to determine.

Memory-to-memory DMA not working
Memory-to-memory DMA completed correctly

The INIT program attempts to test the hard disk DMA circuitry by using it to perform memory-to-memory DMA. The first message indicates an error and will be printed if the WDI hard disk controller DMA is working incorrectly. The second message will be printed informationally if DMA is working correctly as far as the INIT program is able to determine.

No index pulses being received

This error message will be printed if the program receives no index pulses from the drive during the test phase of initialization. This is generally an indication that the drive is not rotating or is improperly connected to its controller.

Index pulses being received correctly
Rotational speed: xxxx RPM

This informational message indicates that index pulses are being received from the drive. It also reports the drive's rotational speed in revolutions per minute.

Rotational speed: overflow
Illegal drive speed (must be xxxx RPM +/- yy%)

These error messages indicate that the rotational speed calculated by the INIT program is out of the legal range for this type of drive. This is generally an indication of a malfunction of the drive.

ZPU clock must be set to 4MHz for correct operation of hard disk

This error message is printed if the operator is attempting to use a hard disk in a 2MHz (instead of a 4MHz) computer system. The problem can be corrected by switching the ZPU to 4MHz operation.

Incorrect operation of WDI and hard disk

This error message is printed following a number of the above errors to indicate to the operator that there is a problem with the operation of either the hard disk or the WDI hard disk controller.

Read error: alternate track register
Do you wish to format drive anyway (Y/N)?

A list of alternate tracks for the hard disk (to be used in cases of hard errors on any normal data tracks) is stored for use by the operating system in a dedicated area of track 0 (cylinder 0, surface 0). This area is called the alternate track register or alternate track table. The INIT program will attempt to read this register prior to initializing the drive so that the alternate tracks which have already been declared can be redeclared following the initialization. This error message and question is displayed if the INIT program is unable to read the alternate track register. Answering **no** to this question will abort the INIT program and allow the problem to be corrected. Answering **yes** to this question will tell the INIT program to attempt to format the drive. In this latter case, the alternate track register information will be lost.

Cannot be assigned an alternate track

Since the alternate track register is stored on track 0 (cylinder 0, surface 0), this one track can never have an alternate track assigned to it. An attempt to do so will result in the above error message.

INIT ERROR MESSAGES FOR CDOS

The following error messages may appear during the INIT program while running under CDOS only:

Warning: This CDOS has been GENerated for single sided operation only

Warning: This CDOS has been GENerated for single density operation only

These are messages warning that the operator has specified a type of diskette which does not match the CDOS configuration. The specified initialization will still be performed correctly provided the user has the correct hardware. For example, a user may have used CDOSGEN to create a CDOS capable of accessing single sided or single density diskettes, yet still have hardware which can format double sided or double density diskettes.

Bitmap written to disk
Warning: Execute STAT to correct bitmap if files still exist

If the INIT program is used to initialize a portion of a hard disk (for example, to format only the alternate track region of the disk), files might still be left on the disk. The hard disk space allocation map (called the bitmap) stored on the disk must be updated following initialization. This is done by executing the STAT program on that same disk following execution of INIT.

Disk name (up to 8 characters)?
Date on disk (mm/dd/yy)?
Number of directory entries (64-512)?

These questions will be asked by the INIT program following formatting of CDOS disks only. They are used to determine the disk directory label, which tells CDOS how to access the files on that disk. They may be answered with their default values (displayed in angle brackets, <>, following the questions) by pressing RETURN. New responses may be given by typing them and pressing RETURN.

RDOS 2.52

Version 02.52 of Cromemco RDOS is available and affects CDOS with several enhancements. Briefly, this latest version allows CDOS users to boot from any floppy disk drive. A change was made to the user entry jump table in RDOS to give ROM-based, real time application programs the ability to boot from any of the four drives. See **Application Note: RDOS 2.52**, part number 023-9042, for details.

CDOS TRACKS

Two new system calls, specified elsewhere in this note, read and write tracks on CDOS format diskettes and hard disks. The following table gives the track size of different possible media.

Disk/ Diskette	Density	Which Tracks	Bytes Per Sector	Sectors	Track Size (in bytes)
Small (5")	Single	All Tracks	128	18	2304
Small (5")	Double	First Track	128	18	2304
		Other Tracks	512	10	5120
Large (8")	Single	All Tracks	128	26	3328
Large (8")	Double	First Track	128	26	3328
		Other Tracks	512	16	8192
Hard (8")		All Tracks	512	20	10240
Hard (5")		All Tracks	512	20	10240

The track size is determined by multiplying the number of bytes per sector, usually 128 or 512 bytes, by the number of sectors in the track. For example, the first track of a small diskette which is formatted for double density has a track size of 2304 bytes. The remaining tracks all have a track size of 5120 bytes.

HARD DISK IDENTIFICATION

Cromemco now has three different hard disks available for system use. This increase has lead to an allotment of 24 bytes in the first sector as the disk specifier. Cromemco software continues to support existing disks identified as HD11SD and CH11SD. These two identifications are for the CDOS and Cromix Operating Systems on the old style 11 megabyte hard disk. However, the SD identification will be dropped and all new disks initialized for this model will have the new identifications H8-1 and C8-1 respectively. A second 8" hard disk with more tracks is identified as H8-2 and C8-2 for the CDOS and Cromix Operating Systems, respectively. The third type of hard disk is the 5" drive with CDOS and Cromix Operating System designations of H5-1 and C5-1 respectively.

The decimal locations of the 24 bytes in the first sector are:

104 - 105	Number of cylinders (2 bytes)
106 - 107	Number of alternate tracks (2 bytes)
108	Number of surfaces (1 byte)
109	Number of sectors per track (1 byte)
110 - 111	Number of bytes per sector (2 bytes)
112 - 113	Byte count of start of alternate track table (2 bytes)
114 - 115	Cylinder number of start of disk (2 bytes)
116 - 119	Reserved for future use (4 bytes)
120 - 123	Hard disk identifier (4 bytes)
124 - 127	Reserved for future use (4 bytes)

FILE CONTROL BLOCK (FCB) CHANGES

An important change in CDOS is that FCB's can be 33 or 36 bytes long. The random record number is determined by a 24 bit value in the 3 bytes at the end of an FCB. The addresses are bytes 33, 34, and 35, counting from zero. The three bytes are least significant byte (LSB), most significant byte (MSB), and file size byte (FSB) respectively. The FSB is not accessed except by system call 35 to determine file size. FSB is normally zero, as nonzero values indicate overflow past end of file.

LSB and MSB contain the value of the record to read. This value is from 0 to 65536 and can access up to an 8 megabyte file. All records are 128 bytes.

NEW SYSTEM CALLS

CDOS system calls are designed expressly for the CDOS Operating System. However, Cromix Operating System users can execute programs written for the CDOS system by using the Cromix CDOS Simulator, **sim.bin**, which is supplied with the Cromix Operating System. **Sim.bin** executes a subset of the CDOS system calls for Cromix Operating System users and never does more than the CDOS Operating System already does.

Each CDOS system call below describes the action sim.bin takes. There are three different actions which Sim.bin may take. First, the system call may be executed just as expected under the CDOS system. This is referred to as **implemented in sim.bin.** Second, the system call may be illegal and will cause errors and error messages to be generated. This is referred to as **illegal in sim.bin.** Third, the system call may not execute under **sim.bin** with no

error messages generated. It is as though **sim.bin** ignores the system call and proceeds executing the remaining program statements. This is referred to as **ignored by sim.bin.** All calling and return parameters refer to Z80 registers A, B, C, D, E, H, and L.

Read Track and Write Track

These two new system calls allow reading and writing of entire tracks on a floppy or hard disk. **Read track** (0A0h) and **write track** (0A1h) work the same way as **read** and **write logical blocks,** except that they read and write complete tracks. These calls access all of the data blocks (sectors) on the track. They do not access the sector header, sector trailer, and formatting information. The following is a brief description of these calls.

System call:	**Read Track** 160 (0A0h)
Purpose:	This call is used to read data on a single track of a floppy diskette or hard disk.
Calling Parameters:	B = the disk number. 1 = A, 2 = B, etc. If bit 7 is set, CDOS will read interleaved on the track. If bit 7 is reset, CDOS will read sequentially on the track. Reading sequentially is recommended in most cases. DE = track number. L = surface number.
Return Parameters:	None.

Home the drive before the first **read track** system call and after a sequence of read track system calls. These calls execute at the same speed when accessing a hard disk whether or not interleaved mode is used. This call executes considerably slower in interleaved mode than noninterleaved mode when accessing a floppy diskette. It is recommended that programs use noninterleaved mode.

This call is illegal in the Cromix CDOS Simulator (**sim.bin**).

System call: **write track**
(0A1h)

Purpose: This call is used to write data on a single track of a floppy diskette or hard disk.

Calling parameters: B = the disk number. 1 = A, 2 = B, etc. If bit 7 is set, CDOS will write interleaved on the track. If bit 7 is reset, CDOS will write sequentially on the track.

DE = track number.

L = surface number.

Return parameters: None.

Home the drive before the first **write track** system call and after a sequence of write track system calls. This call executes at the same speed when accessing a hard disk whether or not interleaved mode is used. It executes considerably slower in interleaved mode than noninterleaved mode when accessing a floppy diskette. It is recommended that program uses noninterleaved mode.

This call is illegal in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Deselect Disk**
(0A2h)

Purpose: Deselects the current disk.

Calling parameters: None.

Return parameters: None.

This is the same call as the old System Call number 12 (0CH) as described in the June 1981 **CDOS Instruction Manual**, part number 023-0036. The old system call has been redefined as explained below.

ADDITIONAL NEW CDOS SYSTEM CALLS

Several additions make the current version of CDOS capable of executing many programs written for CP/M* version 2.2 and below. These 15 new CDOS calls are described below.

System Call:	**Direct Console I/O** 6 (06h)
Purpose:	This call allows console input without echo and console output without checking.
Calling parameters:	**E** contains -1 (0FFh) for input or an ASCII character for output.
Return parameters:	**A** returns a value only if the calling parameter E was -1 (0FFh). This value is an ASCII character representing the character typed on the keyboard. If no character is ready for input, the value in **A** is 0 (00h).

This call ignores potentially important I/O such as CNTRL-C, CNTRL-P, CNTRL-S, and other control characters.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

* CP/M is a trademark of Digital Research.

System Call: **Return Version Number**
12 (0Ch)

Purpose: This call returns the value 0202h.

Calling parameters: None.

Return parameters: **HL** contains the number.

This call replaces the old system call 12, named **deselect disk**, which has become the CDOS system call 0A2H. See System Call **Deselect Disk**, above.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Write Protect Disk**
28 (1Ch)

Purpose: This call allows some programs written for CP/M and CP/M-like systems to run without modifications under CDOS.

Calling parameters: None.

Return parameters: None.

This call is ignored by the Cromix CDOS Simulator (**sim.bin**).

System Call: **Get Read-only Vector**
29 (1Dh)

Purpose: This call allows some programs written for CP/M and CP/M-like systems to run without modifications under CDOS.

Calling parameters: None.

Return parameters: **HL** register pair contains a 00, meaning null operation.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Set File Attributes**
30 (1Eh)

Purpose: This call allows some programs written for CP/M and CP/M-like systems to run without modifications under CDOS.

Calling parameters: None required. Registers DE may contain the FCB address.

Return parameters: None.

This call is ignored by the Cromix CDOS Simulator (**sim.bin**).

System Call: **Illegal**
31 (1FH)

Purpose: This call is illegal under CDOS.

Calling parameters: None.

Return parameters: None.

This call returns an error message and is illegal under the Cromix CDOS Simulator (**sim.bin**).

System Call: **Set/Get User Code**
32 (20h)

Purpose: This call allows a program to determine or interchange the currently active user number.

Calling parameters: E contains FF to get the user code. Any other value in E will set the user code to the value in E.

Return parameters: E contains the user code if a get user call was made.

This call will allow some programs written for CP/M to set or get the user code without modification under CDOS.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Read Random Record**
33 (21h)

Purpose: This call allows programs to read a random file.

Calling parameters: **DE** contains the FCB address.

Return parameters: **A** contains the Return code where the code value below on the left has the associated meaning on the right.

01	=	reading unwritten data
02	=	not returned
03	=	cannot close current extent
04	=	seek to unwritten extent
05	=	not returned by this call
06	=	seek past physical end of disk

The read random record system call performs the read operation with a particular random record number. (See the Discussion under **FILE CONTROL BLOCK.**)

In order to process a file with this system call, first open the base extent which is extent 0. Doing so insures the file is properly recorded in the directory and is visible in DIR

requests. The record number is not advanced as in the case of sequential reads. Repeated random reads read the same record. Since each random read operation sets the logical extent and current record values, the file can be sequentially read or written after any random read operation.

In switching from random to sequential read the last random record will be reread. Avoid this side effect by advancing the random read position before the sequential operation.

Error codes 01 and 04 refer to random read calls which access unwritten data blocks or uncreated extents. Error 03 does not normally occur, but can be cleared by reopening or rereading extent zero. Error 06 occurs if FSB is nonzero.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Write Random Record**
34 (22h)

Purpose: This call allows programs to write to random files.

Calling parameters: **DE** contains the FCB address.

Return parameters: **A** contains the return code.

Data is written to the disk from the current disk buffer.

The random record number is not changed as a result of this system call. Random writes and random reads may be interchanged but they will act on the same record until the record number is changed by the program. Repeated random writes will overwrite the same record. Writing or reading the final record in an extent will not cause an automatic extent switch as it does in sequential writing.

The return codes are the same as in **READ RANDOM** explained above with the inclusion of code number 05. This code means that a new extent was not created to prevent directory overflow.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Calculate file size**
35 (23h)

Purpose: This call calculates the size of a file.

Calling parameters: **DE** contains the FCB address.

Return parameters: Random Record Field set in the last 3 bytes of FCB (see discussion above).

This call computes the size of a file by having the DE register pair address an FCB in random mode format. Programmers can append data to the end of an existing file by a system call 35, which sets the random record position to the end of file. A sequence of random write system calls will append data to the file. If a random file has missing records, the file size will reflect the total of missing and existing records. The maximum file size is 65536.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Set Random Record**
36 (24h)

Purpose: This call produces the random record position of the file which is being sequentially written or read.

Calling parameters: **DE** contains FCB address.

Return parameters: Random record field set in the last 3 bytes of FCB (see discussion above).

This call allows programs to sequentially scan a file and periodically save **keys** for later random access. This system call can aid the transition from sequential access to random access.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

System Call: **Not Used**
37 (25h)

Purpose: This call allows some programs written for CP/M and CP/M-like systems to run with no modifications under CDOS.

Calling parameters: None.

Return parameters: None.

This system call is ignored by the Cromix CDOS Simulator **(sim.bin)**.

System Call: **Illegal**
38 (26h)

Purpose: Not used.

Calling parameters: None.

Return parameters: None.

This system call is illegal under the Cromix CDOS Simulator **(sim.bin)**.

System Call: **Illegal**
39 (27h)

Purpose: Not used.

Calling parameters: None.

Return parameters: None.

This system call is illegal under the Cromix CDOS Simulator **(sim.bin)**.

System Call: **Write Random Record With Zero Fill**
40 (27h)

Purpose: This call zeros out clusters before writing the first random record.

Calling parameters: **DE** contains the FCB address.

Return parameters: **A** contains the return code.

This call simply fills a cluster with zeros before the first random record is written into the cluster.

This call is implemented in the Cromix CDOS Simulator (**sim.bin**).

SPELLMASTER USERS

CDOS has grown in size with the inclusion of the many features discussed in this note. The increased size of CDOS can prevent some SpellMaster software users from using their existing user dictionaries which have grown with the addition of new words.

Users for whom this is the case have several possible courses of action:

1. The recommended course is to upgrade to CDOS version 02.52 and start with a fresh **spellcds.usr** dictionary.

2. Continue to use CDOS version 02.36.

3. Attempt to configure the smallest CDOS possible by running CDOSGEN and specifying only those disk drives which exist in the system. This course is not recommended by Cromemco.

KNOWN PROBLEMS

Some programs which ran under previous versions of CDOS **will not** run under CDOS version 02.52. This is because the new version requires more space in RAM memory than previous versions. Particularly affected are large CDOS 02.52 configurations with multiple hard disks. Cromemco Software packages which are also affected are:

Data Base Management System
Data Base Reporter
SpellMaster Proofreading Program
(See discussion above)

Other software packages, manufactured by companies other than Cromemco, may also encounter memory space problems.

Users may run the above mentioned Cromemco programs under earlier versions of CDOS without difficulty. Before using any software with CDOS version 02.52, users should verify that the software can be executed in the available memory space. This can be done by running the STAT utility to determine the size of the CDOS **operating system.** The **user memory size** is the maximum size allowed for any user program.

VERSION NUMBER SUMMARY

@	version 02.01
CDOS	version 02.52
CDOSGEN	version 02.52
DUMP	version 02.02
EDIT.COM	version 00.10
INIT	version 02.71
RDOS	version 02.52
SCREEN.COM	version 01.24
STAT	version 02.19
WRTSYS.COM	version 02.01
XFER	version 02.05